(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平10－132594

(43)公開日　平成10年(1998)５月22日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０１Ｃ　21/00		Ｃ０１Ｃ　21/00	Ｃ
Ｇ０８Ｇ　1/0969		Ｃ０８Ｇ　1/0969	
Ｇ０９Ｂ　29/10		Ｃ０９Ｂ　29/10	Ａ

審査請求　未請求　請求項の数６　ＯＬ　（全　６　頁）

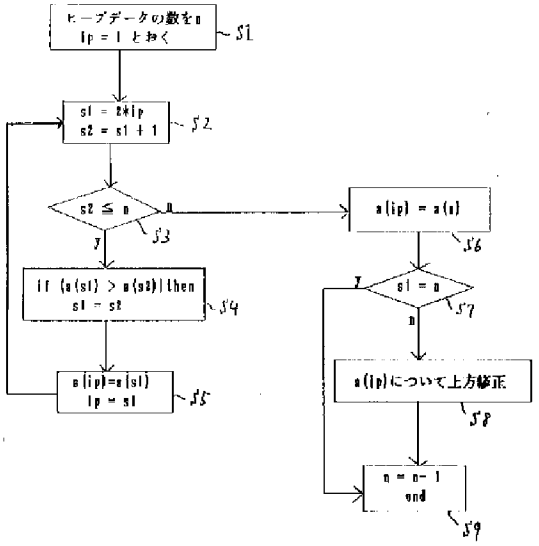
(21)出願番号　特願平8－290440

(22)出願日　平成８年(1996)10月31日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号
(72)発明者　菊池　敦
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(74)代理人　弁理士　佐藤　隆久

(54)【発明の名称】　経路探索方法

(57)【要約】
【課題】　現在地から目的地までの経路探索を高速に行うことができる経路探索方法を提供する。
【解決手段】　地図上の交差点に対応した各ノードのラベル（距離）をヒープで管理しながら、ダイクストラ法を用いて経路探索を行う。このとき、ラベルを確定する際に、前記ヒープのルートと最後の要素を交換し、当該交換後の最後の要素を前記ヒープから切り離した後に、当該ヒープの２番目の要素と３番目の要素とで値（ラベル）を比較し、値が小さい要素をルートに自動的に移動し、当該移動により抜けた位置が親の位置となる子の要素相互間で値を比較し、値が小さい要素を前記移動により抜けた位置に自動的に移動して前記ヒープの修正を行う。



【特許請求の範囲】
【請求項１】記録媒体から読み出した地図情報に含まれる各交差点情報をノードとし、ダイクストラ法に基づいて、現在地に最も近いノードである始点と目的地に最も近いノードである終点との間で、探索すべきノードのラベルをヒープを用いて管理しながら、始点からの距離が最少な点からラベルを確定して目的地までの経路を探索する経路探索方法において、
前記ラベルを確定する過程で、前記ヒープのルートと最後の要素を交換し、当該交換後の最後の要素を前記ヒープから切り離した後に、
当該ヒープの２番目の要素と３番目の要素とで値（ラベル）を比較し、値が小さい要素をルートに自動的に移動し、当該移動により抜けた位置が親の位置となる子の要素相互間で値を比較し、値が小さい要素を前記移動により抜けた位置に自動的に移動して前記ヒープの修正を行う経路探索方法。
【請求項２】前記抜けた位置が親となる子の要素が存在しない場合には、一番最後の要素を当該抜けた位置に移動し、親の要素の値が子の要素の値以下であるという関係が成り立つように、前記最後に移動した要素について上方修正を行う請求項１に記載の経路探索方法。
【請求項３】前記現在地は衛星からの信号に基づいて決定され、前記目的地は入力され、
前記ラベルを確定して得られた経路を出力する請求項１に記載の経路探索方法。
【請求項４】記録媒体から読み出した地図情報に含まれる各交差点情報をノードとし、ダイクストラ法に基づいて、現在地に最も近いノードである始点と目的地に最も近いノードである終点との間で、探索すべきノードのラベルをヒープを用いて管理しながら、始点からの距離が最少な点からラベルを確定して目的地までの経路を探索する経路探索方法において、
前記ラベルを確定する過程で、前記ヒープのルートと最後の要素を交換し、当該交換後の最後の要素を前記ヒープから切り離した後に、
当該ヒープの２番目の要素と３番目の要素とで値（ラベル）を比較し、値が大きい要素をルートに自動的に移動し、当該移動により抜けた位置が親の位置となる子の要素相互間で値を比較し、値が大きい要素を前記移動により抜けた位置に自動的に移動して前記ヒープの修正を行う経路探索方法。
【請求項５】前記抜けた位置が親となる子の要素が存在しない場合には、一番最後の要素を当該抜けた位置に移動し、親の要素の値が子の要素の値以上であるという関係が成り立つように、前記最後に移動した要素について上方修正を行う請求項４に記載の経路探索方法。
【請求項６】前記現在地は衛星からの信号に基づいて決定され、前記目的地は入力され、
前記ラベルを確定して得られた経路を出力する請求項４に記載の経路探索方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、現在地から目的地までの最短経路を高速に求めることができる経路探索方法に関する。
【０００２】
【従来の技術および発明が解決しようとする課題】地図上に車両の走行位置を表示するカー・ナビゲーションが知られている。このカー・ナビゲーションは、３つ以上の衛星からの信号を受信して、車両の現在位置を測定するＧＰＳ(Global Positioning System) 機能と、地図情報を収めたＣＤ－ＲＯＭから車両の走行位置周辺の地図を検索して表示する機能とを備えている。ところで、近年、カー・ナビゲーションにおいて、車両の現在位置から目的地に至るまでの最短（最適）経路を求めて表示する機能の開発が行われている。
【０００３】このように、現在地から目的地までの最短距離を求めるときに例えば、２点間の最短経路を求めるダイクストラ法が用いられる。このダイクストラ法では、地図上の交差点をノードとし、現在地から目的地までの最短経路を、目的地に最も近いノードを計算開始ノードとし、現在地に最も近いノードを計算終了ノードとし、計算開始ホードから計算終了ノードまでを含む道路地図内における計算開始ノードと計算終了ノードとの間の距離に応じたリンクコストを計算して求める。このために、計算開始ノードから計算終了ノードに至るリンクコストを順次加算してリンクツリーを構成し、計算終了ノードに到達する最もリンクコストの少ない（距離が最小の）経路を選択する。
【０００４】このようなダイクストラ法による経路探索を行う場合には、距離が最小な点を求めるためのソートプログラムが用いられる。ところで、従来のカー・ナビゲーションのソートアルゴリズムとしては、例えばヒープソートが用いられている。しかしながら、カー・ナビゲーションでは、経路探索に必要とされる時間が長いという問題がある。
【０００５】本発明は、上述した従来技術に鑑みてなされ、現在地から目的地までの経路探索を高速に行うことができる経路探索方法を提供することを目的とする。
【０００６】
【課題を解決するための手段】ところで、ヒープソートは大きく分けると次の２つの部分からなる。
（Ａ）：初期ヒープを作る
（Ｂ）：交換と切り離しにより崩れたヒープを正しいヒープに直す
上記（Ｂ）は、以下に示す（Ｂ１）～（Ｂ３）の３つの部分からなる。
（Ｂ１）：Ｎ個のヒープデータがあったとき、ルート（木の根に相当する）の値は最小値になっている。この

ルートと最後の要素（Ｎ）を交換し、最後の要素（交換前のルートの値）を木から切り離す。

【０００７】（Ｂ２）：（Ｂ１）により、Ｎ－１個のヒープが構成されるが、ルートのデータがヒープの条件を満たさない。そこで、ルートのデータを下方移動してヒープを修正し、正しいヒープを作る。

（Ｂ３）：Ｎ－１個のヒープについて、ルートと最後の要素（Ｎ－１）を交換し、最後の要素（交換前のルートの値）を木から切り離す。

以上を繰り返していけば、Ｎ，Ｎ－１，Ｎ－２，．．に小さい順にデータが確定されるとともに、ヒープのサイズが１つずつ小さくなっていき、最後に整列が終了する。

【０００８】ここでヒープとは何かについて説明する。図３に示すように、全ての親が必ず２つの子を持つ（最後の要素は左の子だけでもよい）完全２分木で、どの親と子をとっても、親≦子になっている木をヒープという。なお、左の子と右の子の大小関係は問わない。このヒープは、２分木をレベルごとの走査で得られる順に配列に格納して表すこともできる。

【０００９】左の子の位置をＳとすると、右の子の位置はＳ＋１となる。このとき２つの親の位置ＰはＳ／２で求められる。新しいデータをヒープに追加するには次のように行う。この操作は空のヒープから始めることもできる。

（１）新しいデータをヒープの最後の要素として格納する。

（２）その要素を子とする親のデータと比較し、親の方が大きければ子と親を交換する。

（３）次に、その親を子として、その上の親と同じことを繰り返す。繰り返しの終了条件は、子の位置がルートまで上がるか、全てにおいて親≦子となることである。このように新しいデータをヒープの関係を満たすまで上方に上げていくことを上方修正（移動）という。１つのデータをヒープに追加するとき、比較は最高「$\log_2 N$」回だけ行われ、このときのメモリ内でのデータ移動回数は「$3\log_2 N$」回である。

【００１０】ところで、前述した（Ｂ２）に示すヒープ修正法では、Ｎ番目の要素を入れたところ（最初はルートのところ）が、ヒープがこわれているため、そこを修正する。すなわち、この要素の子に当たる要素を比較して小さい方の要素とＮ番目の要素を比較し、ヒープの条件が満足されていなければ交換する。この交換した場所を親としたときの子について同じことを繰り返し、下方移動による修正を行う。この下方移動による修正は、比較の回数が最高$2\log_2 N$になりメモリ内でのデータ移動回数は$3\log_2 N$になる。

【００１１】ここで、本出願の発明者は、前述した（Ｂ２）に示すヒープ修正法では、一番後ろのＮ番目の要素をルートに入れるために、無駄な比較演算を多数回行っていることを見い出した。すなわち、ヒープの最後の要素の値は大きなものである可能性が高いため、この最後の要素は殆どの場合において一番後ろ近くまで下方修正される。従って、このヒープの最後の要素が次のルートになる可能性は無い。そして、この下方修正の過程において、当該最後の要素が大きいと判断されることが明らかな比較演算が何度も行われる。

【００１２】本発明の経路探索方法は、記録媒体から読み出した地図情報に含まれる各交差点情報をノードとし、ダイクストラ法に基づいて、現在地に最も近いノードである始点と目的地に最も近いノードである終点との間で、探索すべきノードのラベルをヒープを用いて管理しながら、始点からの距離が最少な点からラベルを確定して目的地までの経路を探索する経路探索方法であって、前記ラベルを確定する過程で、前記ヒープのルートと最後の要素を交換し、当該交換後の最後の要素を前記ヒープから切り離した後に、当該ヒープの２番目の要素と３番目の要素とで値（ラベル）を比較し、値が小さい要素をルートに自動的に移動し、当該移動により抜けた位置が親の位置となる子の要素相互間で値を比較し、値が小さい要素を前記移動により抜けた位置に自動的に移動して前記ヒープの修正を行う。

【００１３】また、本発明の経路探索方法は、記録媒体から読み出した地図情報に含まれる各交差点情報をノードとし、ダイクストラ法に基づいて、現在地に最も近いノードである始点と目的地に最も近いノードである終点との間で、探索すべきノードのラベルをヒープを用いて管理しながら、始点からの距離が最少な点からラベルを確定して目的地までの経路を探索する経路探索方法であって、前記ラベルを確定する過程で、前記ヒープのルートと最後の要素を交換し、当該交換後の最後の要素を前記ヒープから切り離した後に、当該ヒープの２番目の要素と３番目の要素とで値（ラベル）を比較し、値が大きい要素をルートに自動的に移動し、当該移動により抜けた位置が親の位置となる子の要素相互間で値を比較し、値が大きい要素を前記移動により抜けた位置に自動的に移動して前記ヒープの修正を行う。

【００１４】

【発明の実施の形態】以下、本発明の実施形態に係わるカー・ナビゲーション方法について説明する。図１は、本実施形態に係わるカー・ナビゲーション装置２１の構成図である。図１に示すように、カー・ナビゲーション装置２１は、ＧＰＳ受信部２３、ＣＤ－ＲＯＭドライバ２５、キーボード２９表示部３１、スピーカ３３およびデータ処理部３５を有する。ＧＰＳ受信部２３は、ＧＰＳ(Global Positioning)により自動車などの移動体の位置を検出するもので、例えば４個のＧＰＳ人工衛星３９から地球に無線送信される位置検出用の航法電波を受信して、車両の現在位置を検出する。

【００１５】ＣＤ－ＲＯＭドライバ２５は、地図情報記

録媒体であるＣＤ－ＲＯＭ２７がセットされると、これを駆動して道路地図の情報やそれに関連する情報を読みだす。キーボード２９は、ユーザの操作に応じて、出発地点と目的地点などを入力する。データ処理部３５は、ＣＰＵ４５ａ、ＲＯＭ３５ｂ、ＲＡＭ３５ｃおよび経路探索器３７を有する。

【００１６】カー・ナビゲーション装置２１では、ユーザによる操作によってキーボード２９から目的地が入力されると、ＣＤ－ＲＯＭドライバ２５からの地図情報に基づいて、データ処理部３５の経路探索器３７が後述するように最短経路を生成する。そして、ＣＰＵ３５ａは、この最短経路に基づいて、ナビゲート情報を表示部３１に表示させると共に、スピーカ３３からナビゲート情報に応じた音声を出力させる。経路探索器３７は、ＧＰＳ受信部２３から始点（車両の現在地）と、ユーザによるキーボード２９の操作によって終点（目的地）とが与えられ、ダイクストラ法を用いて始点からの各ノードまでの最短距離を求めるが、この過程で、以下に示す改良ヒープソートを用いる。

【００１７】具体的には、経路探索器３７は、ＣＤ－ＲＯＭドライバ２５から入力した地図情報に基づいて、最初に全てのノード（交差点）までの距離（ラベル）を∞とする。次に、始点のラベルを０とする。始点を探索点として探索を開始する。この場合の探索とは、探索点とつながっている全ての点について、次の操作を行うことである。探索点のラベルと探索点から探索点につながっている点までの距離を加え、その距離が今までについていたラベルより小さければ、ラベルを書き換え、探索すべき点の集合に加える。探索点につながっている点の全てについて探索が終わったら、探索すべき点の集合のなかのラベルの一番小さい点を選び探索点とし、集合から除き探索を行う。これを繰り返し、終点が探索点に選ばれたとき最短距離が求められたことになる。このアルゴリズムでは、探索すべき点の集合にデータを追加したり、その集合から一番小さいデータを選びそれを取り除くという操作を行う。この操作を行うために、経路探索器３７では、探索すべき点の集合をヒープで管理するために、以下に示す改良ヒープソートを行う。この改良ヒープソートは、ヒープの修正方法が通常のヒープソートとは異なる。

【００１８】以下、経路探索器３７において行われる改良ヒープソートについて説明する。この改良ヒープソートは、前述した通常のヒープソートと同様に以下の２つの工程からなる。

（Ａ）：初期ヒープを作る

（Ｂ）：交換と切り離しにより崩れたヒープを正しいヒープに直す

上記（Ｂ）の部分は、以下の（Ｂ１）～（Ｂ３）の工程からなる。

（Ｂ１）：Ｎ個のヒープデータがあったとき、ルート（木の根に相当する）の値は最小値になっている。このルートと最後の要素（Ｎ）を交換し、最後の要素（交換前のルートの要素）を木から切り離す。

（Ｂ２）：（Ｂ１）により、Ｎ－１個のヒープが構成されるが、ルートのデータがヒープの条件を満たさない。そこで、ルートのデータを下方移動して修正し、正しいヒープを作る。

（Ｂ３）：Ｎ－１個のヒープについて、ルートと最後の要素（Ｎ－１）を交換し、最後の要素（交換前のルートの値）を木から切り離す。

以上を繰り返していけば、Ｎ，Ｎ－１，Ｎ－２，．．の小さい順にデータが確定されるとともに、ヒープのサイズが１つずつ小さくなり、最後に整列が終了する。

【００１９】次に、経路探索器３７における上記（Ｂ２）のヒープ修正処理を詳細に説明する。上記（Ｂ１）において木から最後の要素（交換前のルートの要素）を切り離した後に、上記（Ｂ２）において最終的にルートになるのは、当該交換前のルートの要素の２つの子のどちらかである。従って、経路探索器３７では、これは配列の２番目と３番目の小さい方を１番目のルートに移動し、当該移動によって抜けたところを親とする２つの子のうち小さい方を当該抜けた位置に移動する。これを子がいる限り繰り返す。抜けたところの子がいなくなったら一番最後の要素を移動する。ただし、一番最後の要素で移したところは、親≦子の関係が成り立っているとは限らない。そこで最後に移した要素について上方修正を行う。この改良のためのメモリは特にいらない。このとき、改良修正法の比較の回数は、$\log_2 N$となり、メモリ内におけるデータ移動の回数は$\log_2 N$と最後の要素を移動した後の上方修正によるものとを加えた回数になる。

【００２０】通常のヒープソートと上述した改良ヒープソートとで、乱数を発生させてソートしたときの計算時間（ＣＰＵ）および比較の回数の対比結果を下記表１に示す。

【００２１】

【表１】

	CPU時間		比較の回数	
要素の数	Heap	改良heap	Heap	改良heap
10000	0.4	0.3	239820	143332
20000	0.9	0.6	519017	306126
50000	2.7	1.6	1430014	831553
100000	5.8	3.7	3060102	1762490

【００２２】このように、改良ヒープソートを用いたことで、通常のヒープソートを用いた場合の約２／３の時間で処理を行うことができる。なお、上記表１に示したＣＰＵ時間には、最初に初期ヒープを作る時間も入っているから、ルートの切り離しの後の修正処理時間のみを見れば、通常のヒープソートを用いた場合に比べて、処理時間は２／３以下に短縮される。

【００２３】以下、上述した改良ヒープソートを用いた経路探索器３７の処理を図２に示すフローチャートを用いて説明する。

ステップＳ１：ヒープを構成する要素の数をｎとする。経路探索器３７は、ルートａ（１）を引き離し，そのルートに要素を移動するため、その移動する場所ｉｐを１とする。

【００２４】ステップＳ２：経路探索器３７は、移動する場所ｉｐについて、当該ｉｐを親とした子の位置で、左の子と右の子の２つの子の配列の位置を計算する。

【００２５】ステップＳ３：経路探索器３７は、ステップＳ２で計算した右の子の位置に要素があるかどうか調べ、要素があればステップＳ４の処理を行い、要素がなければステップＳ５の処理を実行する。

【００２６】ステップＳ４：経路探索器３７は、ステップＳ２で計算した右の子と左の子のどちらの子が小さいかを調べ、当該小さい子の位置をｓ１とする。

【００２７】ステップＳ５：経路探索器３７は、ステップＳ４において判断された小さい子の要素ａ（ｓ１）をａ（ｉｐ）に移動する。そして、この移動によって空いた位置をｉｐとしてステップＳ２の処理に戻る。

【００２８】ステップＳ６：ステップＳ３において右の子の要素がない場合に実行され、経路探索器３７は、最後の要素ａ（ｎ）をａ（ｉｐ）に移動する。

【００２９】ステップＳ７：経路探索器３７は、最後の要素が、左の子であったかどうかを調べ、そうであればステップＳ９の処理を実行し、そうでなければステップＳ８の処理を実行する。

【００３０】ステップＳ８：ａ（ｉｐ）について、上方修正を行う。

【００３１】ステップＳ９：ヒープの要素の数として、ｎの代わりにｎ－１を用いてステップＳ１～８までの処理を行う。

【００３２】以上説明した経路探索器３７によれば、ダイクストラ法において、探索すべき点の集合なかのラベルの一番小さい点を選ぶ工程に改良ヒープソートを用いることで当該工程を高速化でき、２点間の最短経路を高速に算出することができる。その結果、カー・ナビゲーション装置２１によれば、現在地から目的地までの最短経路を高速に表示部３１に表示すると共に音声による案内をスピーカ３３から出力できる。

【００３３】本発明は上述した実施形態には限定されない。例えば、上述した実施形態では、親≦子の関係が成り立つヒープを用いる場合について例示したが、子≦親の関係が成り立つヒープについても本発明は同様に適用できる。この場合には、経路探索器３７では、木から最後の要素を切り離した後に、配列の２番目と３番目の大きい方を１番目のルートに移動し、当該移動によって抜けたところを親とする２つの子のうち大きい方を当該抜けた位置に移動する。これを子がいる限り繰り返す。抜けたところの子がいなくなったら一番最後の要素を移動する。ただし、一番最後の要素で移したところは、子≦親の関係が成り立っているとは限らない。そこで最後に移した要素について上方修正を行う。

【００３４】また、上述した実施形態では、本発明の経路探索方法をカー・ナビゲーション装置２１に適用した場合に例示したが、本発明は、その他の経路探索にも適用できる。

【００３５】

【発明の効果】以上説明したように、本発明の経路探索方法によれば、ダイクストラ法において探索すべき点の集合なかのラベルの一番小さい点を選ぶ工程に改良ヒープソートを用いることで当該工程を高速化でき、現在地から目的地までの経路を高速に算出することができる。

【図面の簡単な説明】

【図１】図１は、本発明のカー・ナビゲーション装置の構成図である。

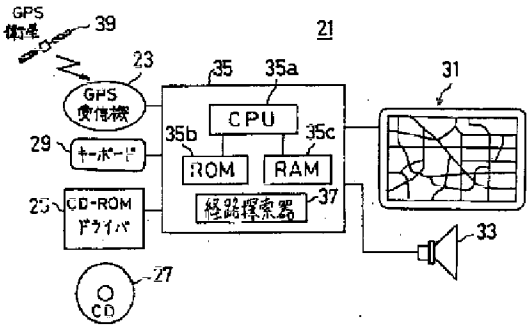
【図２】図２は、図１に示す経路探索器における改良ヒープソート処理を説明するためのフローチャートである。

【図３】通常のヒープソートを説明するための図である。

【符号の説明】

２１…カー・ナビゲーション装置、２３…ＧＰＳ受信部、２５…ＣＤ－ＲＯＭドライバ、２７…ＣＤ－ＲＯＭ、３１…表示部、３３…スピーカ、３５…データ処理部、３５ａ…ＣＰＵ、３５ｂ…ＲＯＭ、３５ｃ…ＲＡＭ、３７…経路探索器、３９…ＧＰＳ衛星

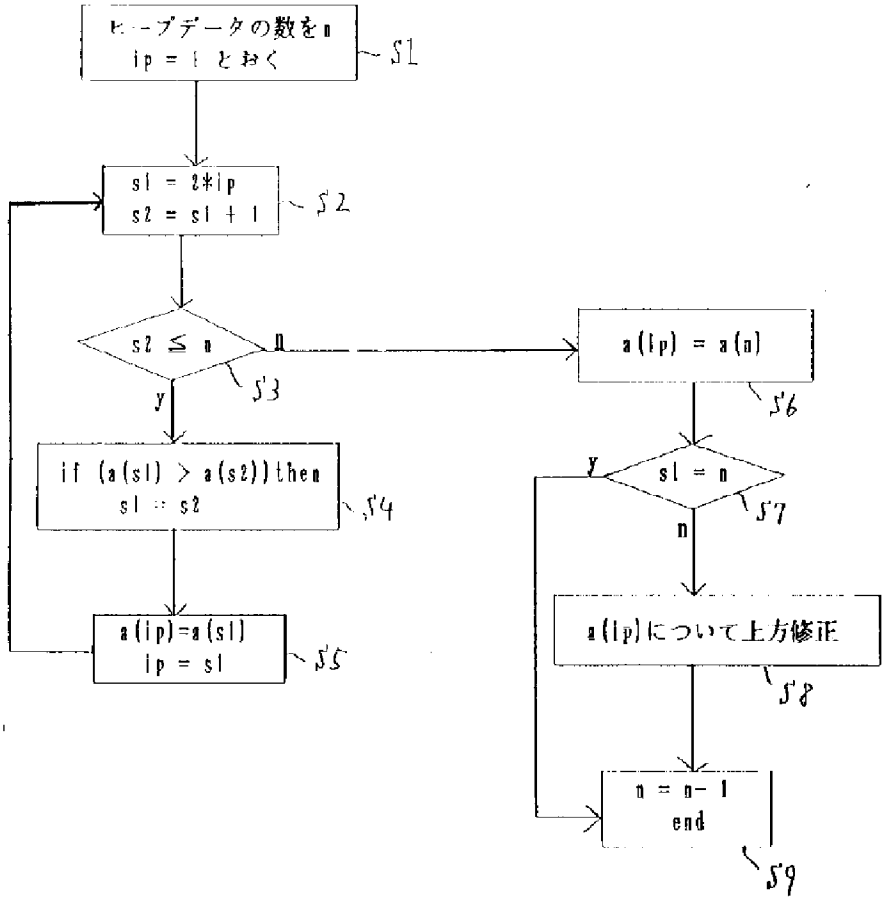
【図1】



【図3】

位置	1	2	3	4	5	6	7	8	9	10	11	12
値	10	25	15	30	26	20	29	35	40	27	28	22
	親	子	子									
		親		子	子							
			親			子	子					
				親				子	子			
					親					子	子	
						親						子

【図2】



L33: Entry 3 of 12 File: JPAB May 22, 1998

PUB-NO: JP410132594A
DOCUMENT-IDENTIFIER: JP 10132594 A
TITLE: ROUTE SEARCH METHOD

PUBN-DATE: May 22, 1998

INVENTOR-INFORMATION:
NAME COUNTRY
KIKUCHI, ATSUSHI

INT-CL (IPC): G01 C 21/00; G08 G 1/0969; G09 B 29/10

ABSTRACT:

PROBLEM TO BE SOLVED: To provide a route search method by which the search for a route from the current position to the destination can be conducted at a high speed.

SOLUTION: While managing a label (distance) of each node by a heap in a manner to correspond to a point of intersection on a map, the Dijkstra's method is used to conduct the search of a route. Then, at the time of determination of the label, a root of the heap is replaced with a final element, and thereafter, the new final element is separated from the heap. Subsequently, a comparison between a second element of the heap and a third element thereof is made on the basis of their values (labels), and the element having a smaller value is automatically moved to the position of the root. This movement of the element creates a vacant position to be occupied by a parent, and a comparison between child elements is made on the basis of their values, and the element having a smaller value is automatically moved to the vacant position to complete modification of the heap.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[The technical field to which invention belongs] this invention relates to the path planning method that the shortest path from a its present location to the destination can be searched for at high speed.
[0002]
[Description of the Prior Art] The car navigation which displays a rolling-stock-run position on a map is known. This car navigation is GPS (Global Positioning System) which receives the signal from three or more satellites, and measures the current position of vehicles. It has the function which searches and displays the map around a rolling-stock-run position as a function from CD-ROM which stored map information. By the way, development of the function displayed in car navigation in quest of the shortest (optimum) path until it reaches [from the current position of vehicles] the destination is performed in recent years.
[0003] Thus, the Dijkstra method which searches for the shortest path for two points when finding the curtate distance from a its present location to the destination is used. In this Dijkstra method, the crossing on a map is made into a node, the node near the destination is made into a calculation start node for the shortest path from a its present location to the destination, the node near a its present location is made into a calculation end node, and it calculates and asks for the link cost according to the distance between the calculation start nodes and calculation end nodes in the road map containing from calculation start Hoad to a calculation end node. For this reason, the link cost from a calculation start node to a calculation end node is added one by one, a link tree is constituted, and the path with least (distance is the minimum) link cost which reaches a calculation end node is chosen.
[0004] When performing the path planning by such Dijkstra method, the sort program for searching for a minimum distance point is used. By the way, as sort algorithm of the conventional car navigation, the heap sort is used, for example. However, in car navigation, there is a problem that the time needed for path planning is long.
[0005] this invention is made in view of the conventional technology mentioned above, and aims at offering the path planning method that the path planning from a its present location to the destination can be performed at high speed.
[0006]
[Means for Solving the Problem] By the way, if a heap sort is roughly divided, it will consist of the following two portions.
(A): (B) which makes an initial heap : the above (B) which changes to the right heap the heap which collapsed by exchange and separation consists of three portions of - (B3) shown below (B1).
(B1) When there are heap data of :N individual, the value of the root (it is equivalent to a wooden root) is the minimum value. The element (N) of this root and the last is exchanged and the last element (value of the root before exchange) is separated from a tree.
[0007] (B-2) Although N-1 heap is constituted by : (B1), the data of the root do not fulfill the conditions of a heap. Then, lower part movement of the data of the root is carried out, a heap is corrected, and the right heap is made.
: (B3) About N-1 heap, the last element (N-1) is exchanged for the root, and the last element (value of the root before exchange) is separated from a tree.
If the above is repeated, while data will be decided by order small to N, N-1, N-2, and .., the size of a heap becomes small one at a time], and, finally alignment is completed.
[0008] Something is explained to be a heap here. As shown in drawing 3 , it is the perfect binary tree in which all parents surely have two children (the last element is only for a left child), and even if it takes which parents and child, the tree which is a parent <= child is called heap. In addition, the size relation between a left child and a right child does not ask. This heap can also store and express a binary tree with an array to the order obtained by the scan for every level.
[0009] If a left child's position is set to S, a right child's position will be set to S+1. At this time, two parents' position P is called for by S/2. For adding new data to a heap, it carries out as follows. This operation can also be begun from an empty heap.
(1) Store new data as an element of the last of a heap.
(2) As compared with the data of the parents who make the element a child, if parents are large, a child and parents will be exchanged.
(3) Next, repeat the same thing as the parents on it by making the parents into a child. A child's position goes up to the root, or the end conditions of a repeat are with a parent <= child and a bird clapper in all.
Thus, it is called upward revision (movement) to raise new data up until it fills the relation of a heap. When adding one data to a heap, as for comparison, only the "log2 highest N" time is performed, and the number of times of data movement within the

memory at this time is a "3log2 N" time.
[0010] By the way, by the heap correcting method shown in (B-2) mentioned above, since the heap has broken, the place (beginning place of the root) into which the Nth element was put corrects that. That is, the element which asks the child of this element is compared and the element of the smaller one is compared with the Nth element, and if the conditions of a heap are not satisfied, it exchanges. The thing same about the child when making this exchanged place into parents is repeated, and the correction by lower part movement is made. As for the correction by this lower part movement, the comparative number of times is set to a maximum of 2 log(s)2 N, and the number of times of data movement within memory is set to 3log(s)2 N.
[0011] Here, by the heap correcting method shown in (B-2) mentioned above, the artificer of this application found out performing the useless comparison operation many times, in order to put the Nth rearmost element into the root. That is, since the value of the element of the last of a heap of possibility of being big is high, in almost all cases, the element of this last is revised downward to near the very back. Therefore, there is no possibility that the element of the last of this heap will become the following root. And in the process of this downward revision, a comparison operation with clear it being judged that the element of the last concerned is large is performed repeatedly.
[0012] The path planning method of this invention makes a node each crossing information included in the map information read from the record medium, and based on a Dijkstra method between the starting point which is a node near a its present location, and the terminal point which is a node near the destination In process in which the distance from the starting point is the path planning method of deciding a label from a minimum point and searching for the path to the destination, and the aforementioned label is decided, managing the label of a node for which it should look using a heap After exchanging the root of the aforementioned heap, and the last element and detaching the element of the last after the exchange concerned from the aforementioned heap The 2nd element of the heap concerned and the 3rd element compare a value (label). A value moves a small element to the root automatically, a value is compared between [of the child from whom the position from which it escaped by the movement concerned turns into parents' position] elements, a value moves to the position which escaped from the small element by the aforementioned movement automatically, and the aforementioned heap is corrected.
[0013] The path planning method of this invention makes a node each crossing information included in the map information read from the record medium, and based on a Dijkstra method moreover, between the starting point which is a node near a its present location, and the terminal point which is a node near the destination In process in which the distance from the starting point is the path planning method of deciding a label from a minimum point and searching for the path to the destination, and the aforementioned label is decided, managing the label of a node for which it should look using a heap After exchanging the root of the aforementioned heap, and the last element and detaching the element of the last after the exchange concerned from the aforementioned heap The 2nd element of the heap concerned and the 3rd element compare a value (label). A value moves a large element to the root automatically, a value is compared between [of the child from whom the position from which it escaped by the movement concerned turns into parents' position] elements, a value moves to the position which escaped from the large element by the aforementioned movement automatically, and the aforementioned heap is corrected.
[0014]
[Embodiments of the Invention] Hereafter, the car navigation method concerning the operation gestalt of this invention is explained. Drawing 1 is the block diagram of the car navigation equipment 21 concerning this operation gestalt. As shown in drawing 1 , car navigation equipment 21 has the GPS receive section 23, the CD-ROM driver 25, keyboard 29 display 31, a loudspeaker 33, and the data-processing section 35. The GPS receive section 23 detects the position of mobiles, such as an automobile, by GPS (Global Positioning), receives the navigation electric wave for position detection by which radio transmission is carried out from four GPS satellites 39 to the earth, and detects the current position of vehicles.
[0015] If CD-ROM27 which is a map information record medium is set, the CD-ROM driver 25 will drive this and will read the information on a road map, and the information relevant to it. A keyboard 29 inputs an origin point, a destination point, etc. according to operation of a user. The data-processing section 35 has CPU45a, ROM35b, RAM35c, and the path planning machine 37.
[0016] With car navigation equipment 21, if the destination is inputted from a keyboard 29 by operation by the user, based on the map information from the CD-ROM driver 25, the shortest path will be generated so that the path planning machine 37 of the data-processing section 35 may mention later. And CPU35a makes the voice according to navigation information output from a loudspeaker 33 while displaying navigation information on a display 31 based on this shortest path. Although a terminal point (destination) is given from the GPS receive section 23 by operation of the starting point (present location of vehicles), and the keyboard 29 by the user and the path planning machine 37 finds the curtate distance to each node from the starting point using a Dijkstra method, it is this process and the improvement heap sort shown below is used for it.
[0017] Specifically, the path planning machine 37 makes infinity first distance (label) to all nodes (crossing) based on the map information inputted from the CD-ROM driver 25. Next, the label of the starting point is set to 0. Search is started by making the starting point into a searching point. The search in this case is performing the next operation about a searching point and all the connected points. The distance to the point which has led to the searching point from the label of a searching point and the searching point is added, and if smaller than the label which the distance attached until now, a label will be rewritten and will be added to a set of the point for which it should search. If search finishes about all the points connected to the searching point, the smallest point of the label in a set of the point for which it should search will be chosen, and it will consider as a searching point, and will search by removing from a set. This is repeated, and when a terminal point is chosen as a searching point, it means that the curtate distance was found. With this algorithm, operation of adding data to a set of the point for which it should search, or choosing the smallest data from the set and removing it is performed. In order to manage a set of the point for which the path

planning machine 37 should be searched in order to perform this operation by the heap, the improvement heap sort shown below is performed. This improvement heap sort differs from a heap sort usual in the correction method of a heap.
[0018] Hereafter, the improvement heap sort performed in the path planning machine 37 is explained. This improvement heap sort consists of the following two processes like the usual heap sort mentioned above.
(A): (B) which makes an initial heap : the portion of the above (B) which changes to the right heap the heap which collapsed by exchange and separation consists of a process of the following - (B1) (B3).
(B1) When there are heap data of :N individual, the value of the root (it is equivalent to a wooden root) is the minimum value. The element (N) of this root and the last is exchanged and the last element (element of the root before exchange) is separated from a tree.
(B-2) Although N-1 heap is constituted by : (B1), the data of the root do not fulfill the conditions of a heap. Then, lower part movement is carried out, the data of the root are corrected, and the right heap is made.
: (B3) About N-1 heap, the last element (N-1) is exchanged for the root, and the last element (value of the root before exchange) is separated from a tree.
If the above is repeated, while data will be decided by N, N-1, N-2, and the small order of .., the size of a heap becomes small one [at a time], and, finally alignment is completed.
[0019] Next, heap correction processing of the above (B-2) in the path planning machine 37 is explained in detail. the above (B1) -- setting -- the element (element of the root before exchange) of a tree to the last -- separation -- finally one of two children of the element of the root before the exchange concerned becomes the root in the above (B-2) behind the bottom Therefore, with the path planning vessel 37, this moves the 2nd of an array, and the 3rd smaller one to the 1st root, and moves the smaller one to the position concerned from which it escaped between two children who make parents the place from which it escaped by the movement concerned. This is repeated as long as there is a child. If the child from whom it escaped stops there being, the very last element will be moved. However, a parent <= child's relation is not necessarily realized the place moved with the very last element. Then, it revises upward about the element moved at the end. Especially the memory for this improvement is not needed. At this time, the number of times of comparison of the improvement correcting method is set to log2 N, and the number of times of data movement in memory turns into the number of times which added what is depended on the upward revision after moving the element of log2 N and the last.
[0020] The machine time (CPU) when generating a random number and sorting by the usual heap sort and the improvement heap sort mentioned above, and the contrast result of the comparative number of times are shown in the following table 1.
[0021]
[Table 1]

	CPU時間		比較の回数	
要素の数	Heap	改良heap	Heap	改良heap
10000	0.4	0.3	239820	143332
20000	0.9	0.6	519017	308128
50000	2.7	1.6	1430014	831563
100000	5.8	3.7	3060102	1762490

[0022] Thus, it can process in time of the abbreviation 2/3 at the time of using the usual heap sort by having used the improvement heap sort. In addition, since time to make an initial heap first is also contained in the CPU time shown in the above-mentioned table 1, if only the correction processing time after separation of the root is seen, the processing time will be shortened by 2/3 or less compared with the case where the usual heap sort is used.
[0023] It explains using the flow chart which shows processing of the path planning machine 37 using the improvement heap sort mentioned above hereafter to drawing 2 .
Step S1: Set to n the number of the elements which constitute a heap. In order that the path planning machine 37 may pull apart root a (1) and may move an element to the root, it sets to 1 the place ip where it moves.
[0024] Step S2: About the place ip where it moves, the path planning machine 37 is the position of the child who made the ip concerned parents, and calculates the position of an array of two children, a left child and a right child.
[0025] Step S3: It investigates whether the path planning machine 37 has an element in the position of the child of the right calculated at Step S2, if there is an element, step S4 will be processed, and if there is no element, processing of Step S5 will be performed.
[0026] Step S4: The path planning machine 37 investigates which child of the child of the right calculated at Step S2, and a left child is small, and sets the position of the small child concerned to s1.
[0027] Step S5: The path planning machine 37 moves a small child's element a (s1) judged in step S4 to a (ip). And it returns to processing of Step S2 by setting to ip the position which was vacant with this movement.
[0028] Step S6: Performing, when there is no element of a right child in Step S3, the path planning machine 37 moves last element a (n) to a (ip).
[0029] Step S7: The path planning machine 37 investigates whether you were a left child, if it is so, it will perform processing of step S9, otherwise, the last element performs processing of Step S8.
[0030] Step S8: Revise upward about a (ip).
[0031] Step S9: As the number of the elements of a heap, use n-1 instead of n, and perform processing to step S1-8.
[0032] According to the path planning machine 37 explained above, in a Dijkstra method, the process concerned can be

accelerated by using an improvement heap sort for the process which chooses the smallest point of the label under set of the point for which it should search, and the shortest path for two points can be computed at high speed. Consequently, according to car navigation equipment 21, while displaying the shortest path from a its present location to the destination on a display 31 at high speed, guidance with voice can be outputted from a loudspeaker 33.

[0033] this invention is not limited to the operation gestalt mentioned above. For example, although illustrated with the operation gestalt mentioned above about the case where the heap of which a parent <= child's relation consists is used, this invention is applicable similarly about the heap of which child <= parents' relation consists. In this case, with the path planning vessel 37, after separating the last element from a tree, the 2nd of an array and the 3rd larger one are moved to the 1st root, and the larger one is moved to the position concerned from which it escaped between two children who make parents the place from which it escaped by the movement concerned. This is repeated as long as there is a child. If the child from whom it escaped stops there being, the very last element will be moved. However, child <= parents' relation is not necessarily realized the place moved with the very last element. Then, it revises upward about the element moved at the end.

[0034] Moreover, although it illustrated with the operation gestalt mentioned above when the path planning method of this invention was applied to car navigation equipment 21, this invention is applicable also to other path planning.

[0035]

[Effect of the Invention] As explained above, according to the path planning method of this invention, the process concerned can be accelerated by using an improvement heap sort for the process which chooses the smallest point of the label under set of the point for which it should search in a Dijkstra method, and the path from a its present location to the destination can be computed at high speed.

[Translation done.]